SYSTEM AND METHOD FOR AUTHENTICATING CLIENTS IN A CLIENT-SERVER ENVIRONMENT

Field of the present invention

[0001] The present invention relates to authentication in general, and in particular to authentication in a client-server environment, and more specifically to authentication of clients in the Internet.

Background of the invention

[0002] Authentication is a procedure of determining whether someone or something is, in fact, who or what it is declared to be. In private and public computer networks authentication is commonly done by the use of logon passwords. Typically, every server maintains its own data persistency in order to store authentication data. Therefore, passwords which are available to the client on one server, may be already blocked by another client on another server. This increases the number of different authentication sets which have to be remembered and maintained by the client. In applications that are distributed over several servers with different user authentication systems (e.g. accessing an application through a portal server where the portal server uses its own user database) the client would have to logon more than once.

[0003] Workarounds to allow single sign-on contain approaches like storing logon data for the application servers on the portal server or the use of centralized user databases like Microsoft's® .NET Passport (http://www.passport.com) or Liberty from the Liberty Alliance (http://www.projectliberty.org). This requires that the client be willing to have personal data stored on a third party site with all the data security issues that come along with this approach. Also if the Passport service should be down one cannot logon to the desired service even if the site one wants to use is available.

[0004] Using a user ID/password set for authentication also has the disadvantage that it results in extra network traffic. On a client request the server has to answer by asking for the login data. Only after this is provided, the originally requested information is sent back to the client (see also Figure 7A below). Finally, passwords can often be stolen, accidentally revealed, or simply forgotten.

[0005] For this reason, Internet business and many other transactions require a more stringent authentication process. The use of digital certificates issued and verified by a Certificate Authority (CA) as part of a public key infrastructure is considered to become the standard way to perform authentication on the Internet.

[0006] Digital signatures enable the recipient (server) to verify the identity of the sender (client) and the origin as well as the integrity of the document. Digital signatures are based on asymmetric cryptographic algorithms. The documents are signed with the private key of the sender. The recipient can take the sender's public key, which is provided to him by a Trusted Third Party, and validate the integrity of the document received.

[0007] The implementation of a digital signature procedure into an already existing password logon infrastructure requires considerable modifications on the server side as well as the client side, e.g. additional card reader with specific security applications. The cost of such implementations in an existing system means means that, for the most part, only new client-server infrastructures will use the digital signature procedure. The existence of those two authentication procedures in the client-server environment has the disadvantage that a client has to check at first whether the destination server is supporting the password logon or the digital signature procedure. Depending on that result the client will use the required authentication process supported by the server. It causes much unnecessary network traffic between client and server since the server application itself finally determines the type of authentication.

[0008] Furthermore, the present digital signature authentication procedures have the disadvantage that several screens between client and server have to be exchanged between client and server until the client can provide its authentication information. This causes much unnecessary network traffic.

Brief summary of the invention

[0009] The existing password/user ID based authentication process is replaced by a new digital signature authentication process in which preferably the first HTTP-request header is extended by the client authentication information independently of the authentication process used by the destination server and without server requesting authentication information. The authentication

information preferably includes the client certificate containing the client public key, signed by certification authority, and preferably a hash value calculated over the HTTP-request header data being sent in the request, and encrypted with the Client's private key. The certificate and digital signature may be added during the creation of the HTTP-request header in the client system itself, or may be added later in a server acting as a gateway, proxy, or tunnel.

[0010] A destination server that does not support the new digital signature authentication process will simply ignore the certificate and digital signature in the HTTP-request header and will automatically initiate its own authentication process. The present invention simplifies the existing digital signature authentication process and concurrently allows the coexistence of different authentication processes without changing the HTTP-protocol or causing unnecessary network traffic.

Brief Description of the Drawings

[0011] The novel features of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives, and advantages thereof, will be best understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figures 1A and 1B show prior art HTTP-client-server environments in which the present invention is preferably used;

Figure 2 shows the basic structure of a typical prior art HTTP-header;

Figure 3 shows the inventive structure of the HTTP-header with the certificate and the digital signature;

Figures 4A to 4D show preferred embodiments to insert the certificate together with the digital signature into the HTTP-request header resulting in the inventive structure of the HTTP-request header;

Figure 5 shows an example of a server-client communication environment using the present invention;

Figure 6 shows a preferred embodiment of the authentication data flow in an client-server environment according to Figure 1 A using the inventive structure of the HTTP-request; and

Figures 7A and 7B show a comparison of the prior art authentication process with the inventive authentication process of the present invention based on an example of a online purchase transaction process.

Detailed Description

[0012] With reference to Figure 1A and Figure 1B, there are depicted client-server environments in which the present invention is preferably used. However it should be noted that the present invention may be used on each client-server environment using communication protocols allowing header extensions without violating normal protocol usage. Therefore, the present invention with its preferred embodiments will be described and explained on the currently mostly known HTTP-protocol.

[0013] The HTTP-protocol (Hypertext Transfer Protocol) is an application level protocol for distributed systems. It is a set of rules for exchanging files (text, graphic, images, sound, video, and other multimedia files). Any web server machine 3 contains a HTTP-daemon or so called HTTP-server 4, a program that is designed to wait for HTTP-requests and handle them when arrive. Furthermore, each client machine 1 contains a web browser or so-called HTTP-client 2, sending requests to web server machine 3. When the browser user enters a request by either opening a web file (typing in a URL) or clicking on a hypertext link, the browser builds an HTTP-request and sends it to the Internet Protocol Address indicated in the URL. The HTTP-server 4 in the destination server machine 3 receives the request and, after processing, the requested file is returned.

[0014] In another client-server environment client 1 is communicating with the server 3 via a gateway, a tunnel, or a proxy-server 5 (see Figure 1B).

[0015] Usually HTTP takes place over TCP/IP (Transmission Control Protocol/Internet Protocol), however HTTP is not dependent on TCP/IP. TCP defines a set of rules to exchange messages with other Internet points at the information package level, and IP defines a set of rules to send and receive messages at the Internet address level.

[0016] An HTTP-request header consists of the HTTP method (GET, HEAD, POST, etc.), the Universal Resource Identifier (URI), the protocol version and optional supplemental information. An HTTP-response consists of a status line, which indicates success or failure of the request, a description of the information in the response (meta information) and the actual information request.

[0017] With respect to Figure 2, there is depicted the basis structure of a prior art HTTP-request header. Each HTTP-request must contain at least a header. Only HTTP-Post requests contain header and body data. Following information are preferably contained in a HTTP-request header:

Resources to be accessed by the HTTP-request (e.g. file, servlet);
The host name of the server (e.g. www.ibm.com);
Browser name and version (e.g. Netscape Version 7.1);
Operating system of the client (e.g. Windows XP); and
Character set that can be understood by the browser (e.g. iso-8859-1).

[0018] Each HTTP-header may include supplemental information not defined by the HTTP-protocol and which does not conflict with existing applications using the HTTP-protocol. That means that an application which uses the HTTP-protocol and which is not configured to process that supplemental information simply ignores that supplemental information without interrupting its execution.

[0019] With respect to Figure 3, there is depicted the inventive structure of a HTTP-request header according to the present invention. Following additional information according to the present invention must be included into the HTTP-request header:

the client certificate containing the public key and signed by a certification authority; and

digital signature calculated over the HTTP-request header and if available HTTP-body (Post).

[0020] The certificate and digital signature can be processed by specific tools on the server. A client certificate is a document distributed by a Trusted Third Party that binds a public key to a specific person. The Trusted Party guarantees that the information contained in the certificate is valid and correct. Certificates are standardized by 509. They should contain the digital signature of the Trusted Third party, the name of the person owning the public key, and the public key itself.

[0021] With respect to Figures 4A to 4D, there are depicted preferred embodiments to insert the client certificate and the digital signature into the HTTP-request header,

[0022] With respect to Figure 4A, there is depicted a first embodiment of the present invention to insert Client's certificate 16 together with the digital signature 18 into the HTTP-request header 12. The client system 1 contains a browser 2 with signature capabilities. The browser 2 generates a HTTP-request header 12, accesses the client's private key which is securely stored on a local file system, encrypts a hash value generated over the HTTP-request header 12 and if available body, with the private key resulting in a digital signature 18. That digital signature 18 together with the Client certificate 16 containing the public key is inserted in the HTTP-request header 12. The extended HTTP-request header 14 is sent to the HTTP-server 4 that initiates the authentication process.

[0023] The authentication component 6 which may be part of the HTTP-server or may be a separate component verifies the client certificate information 16 from the HTTP-request header. Verification can either be done by checking the certificate signature of Certification Authority or comparing it with already known certificates contained in its certification database 9. Using the public key contained in the client certificate 16, the digital signature 18 contained in the HTT-request header 12 is decrypted resulting in a hash value calculated by the client 1. Using the same hash algorithm, the hash value is calculated over the HTTP-request header 12 and body if available. If hash values match verification is completed and the authentication is successful and access to an application 8 is given.

[0024] With respect to Figure 4B, there is depicted a second embodiment of the present invention to insert Client certificate 18 together with the digital signature 16 into the HTTP-request header 12. Now the browser 2 has the functionality to communicate with a smart card 10 via a smart card reader 10. The browser 2 generates a HTTP-request header, establishes communication with the smart card 10, the smart card 10 which contains in its security module a private key and Client's certificate encrypts a hash value generated over the HTTP-header 12 and if available body, with the private key (digital signature), and returns a digital signature 18 together with client's certificate 16 to the browser 2. That digital signature 18 together with Client's certificate 16 containing the public key is inserted in the HTTP-request header 12. The extended HTTP-request header 14 is sent to the HTTP-server 4 that initiates the authentication process by using an authentication component (see description to Figure 4 A)

[0025] With respect to Figure 4C, there is depicted a third embodiment of the present invention to insert Client's certificate 16 together with the digital signature 18 into the HTTP-request header 12. In the third embodiment the client system contains a signature component 20. That component acts as a proxy server running on the same client 1 as the browser 2. The browser 2 is configured to use that proxy server 20. Because of this the browser 2 sends the regularly HTTP-request header 12 to the signature component 20 which then inserts the certificate 16 and digital signature 18 analog to the embodiments as described above. The extended HTTP-request header is sent to the HTTP-server 4 that initiates the authentication process by using an authentication component (see description to Figure 4 A).

[0026] With respect to Figure 4D, there is depicted a fourth embodiment of the present invention to insert Client certificate 18 together with the digital signature 16 into the HTTP-request header 12. In that embodiment the client- request (1a/2a; 1b/2b) is routed via a proxy server 22 having an insertion component 20. The insertion component 20 is communicating with an encryption hardware 24 containing private keys and their assigned certificates, which encrypts a hash value generated over the HTTP-request header 12 and if available body, with the private key (digital signature), and returns digital signature 18 with the client certificate 16 to the insertion component 20 inserting them into the HTTP-request header 12. The extended HTTP-request header 14 is sent to the HTTP-server 4 that initiates the authentication process by using an authentication component (see description to Figure 4 A).

[0027] Because the present invention describes additional header data in the HTTP protocol, all combinations of existing clients and servers that are able to process the additional data in the header can work together. If one of the systems is not enabled to handle additional data everything will work as known today.

[0028] To keep the existing base of billions of installed client browsers, an additional signature software could handle the HTTP extension by acting as a proxy component on the local client machine (see Figure 4C). Within company networks (e.g. intranet), this could even be handled by a central proxy server (Fig 4C). Future versions of Web Browsers may eventually have the functionality described with reference to Fig 4A. This way the transition to the new paradigm can happen over time.

[0029] The digital signature can be created using a signature smart card or any other signature hardware. Also a pure software solution with an encrypted key store on the client computer would be a possible implementation.

[0030] Figure 5 shows an example of a server-client communication environment using the present invention. In this example it is assumed that an application 5 is accessed through a portal server 3. In the state of art this situation is handled by either storing the client's identity data on a server that is accessible by the portal server 3 and the application server 5 (e.g. Microsoft's® .NET Passport) or the identity data for the application server needs to be stored on the portal server 3. Both approaches require the user to have his/her data stored on a third party system which is subject to many security issues.

[0031] By digitally signing the request as explained in Figures 4A to 4D, no server needs to store the user data. The portal server 3 can check the identity of the requester against its user database 4, passing the request along to the application server 5 which can do the same using its user database 6. Client 1a accesses the application server 5 through the portal server 3 while Client 1b can access the application server 5 directly. The application server 5 can use its own user database 6 to retrieve profile information for the user.

[0032] The approach even provides higher security since the application server 5 might want to handle only those requests that passed the portal server 3. In this case the portal server 3 forwards the request and additionally signs it. This enables the application server 5 to verify both

signatures in order to either grant or deny access to its services. Client 1a would gain access to the application server 5 while Client 1b would not be served because its request does not go through the portal server 3.

[0033] With respect to Figure 6, there is depicted the authentication a data flow according to the present invention. Client's browser prepares request to the server 10. In a preferred embodiment of the present invention it will be checked whether signing of the HTTP-request header is switched on 20. If not, Client's browser will send a non-signed request to the server 40 and the server checks whether signing is required 50. If signing is required server will send an error message to the client 50. If signing is not required the server will provide access to the desired information 60.

[0034] If signing is switched on the client's browser inserts certificate and digital signature into the HTTP-request header, and sends that HTTP-request header to the server 30. By appending the extra fields to the HTTP header request header the server is able to retrieve the requester's identity from the certificate (authentication) 35. The Client's certificate contains the requester's name and public key:

[0035] Because it is signed by a trusted authority, the server is able to verify that it is a valid certificate issued by the trusted authority. Verification that the message has been sent by the owner of the certificate is possible, because only the owner of the private key belonging to the certificate can have generated the digital signature value in the HTTP-request header which has been calculated over the HTTP-request header data and can be verified through the use of the public key contained in the certificate. If the authentication has been successful, the server provides access to the requested data 60.

[0036] With respect to Figures 7A and 7B, there are depicted typical scenarios of the exchange of information between Web browser (client) and Web server (server) using the prior art authentication process in comparison with the inventive authentication process of the present invention.

[0037] For example, during a purchase process, the client receives and sends data (e.g. a series of text or html pages or blocks of formatted data like XML) from and to the server representing the online shopping system until the order gets confirmed by a specific data transfer operation

(e.g. HTTP Post). In today's applications, the server issues a request to obtain a user Id and password from the client during this process. The user has to supply these data manually before they are sent to the server by the client application (see Figure 7A).

[0038] In an application corresponding to the present invention (see Figure 7B), the client signs the HTTP-request header data being sent to the server by means of a digital signature. The server easily identifies the client by checking the signature. It is therefore not necessary to request and supply the user Id and password, since every data item transmitted is associated with the user's identity. The server may retrieve stored information for this client and use this information in preparing the data which is to be sent to the client ("personalization", profile creation page). Examples for data used for personalization are the user's address (where should the ordered items be delivered to), the users shopping history, the users shopping cart, the web pages visited during the last sessions etc.

[0039] By checking the identity of the user (which can be done at any time during the flow), the server may find out that the user never visited this site before. The server may then send data containing a request to specify preferences and detailed user data (profile creation page). The user supplies these data, the client application sends it to the server and the server stores these data used for personalization in its data base.

[0040] Since every data transfer is signed, the user ID of the client is known to the server as soon as the client visits the first page. The personalization may therefore take place early during the process.

[0041] When the user chooses to switch off signing, the server recognizes this fact and may send a page containing an indication to switch on signing or might use traditional user ID / password scenarios instead (not shown).